CLAIMS

1. A method for validating programs, the method comprising:

receiving a language-independent description of a computer program, the language-independent description comprising a definition module and an implementation module;

validating the language-independent description;

generating a language-dependent program from the language-independent description, the language-dependent program comprising an interface and a class; and

validating the language-dependent program.

2. The method of claim 1 wherein validating the language-independent description comprises validating the syntax of the definition module and the implementation module.

3. The method of claim 1 wherein validating the language-dependent program comprises compiling the interface and the class.

4. The method of claim1 wherein the definition module and the implementation module are represented in a meta-language or using a tree structure.

5. A method for validating programs, the method comprising:

receiving a language-independent description of a computer program, the language-independent description comprising a definition module and an implementation module;

validating the language-independent description;

generating a language-dependent program from the language-independent description, the language-dependent program comprising a script code section; and

validating the language-dependent program.

6. The method of claim 5 wherein validating the language-dependent program comprises:

extracting language elements from the script code section; and

comparing the extracted language elements with the definition module.

7. The method of claim 6 wherein extracting language elements comprises generating a symbol table from the script code section.

8. The method of claim 5 wherein generating the language-dependent program comprises:

generating language-dependent code comprising an interface and a class.

9. The method of claim 5, wherein validating the language-dependent program comprises:

extracting language elements from the script code section;

comparing the extracted language elements with the definition module;

generating language-dependent code comprising an interface and a class; and

compiling the interface and the class.

10. A method for validating programs, the method comprising:

receiving a language-independent description of a computer program, the language-independent description comprising a definition module and an implementation module;

validating the language-independent description;

generating a first language-dependent program from the language-independent description, the first language-dependent program comprising a first script code section;

generating a second language-dependent program from the language-dependent description, the second language-dependent program comprising a second script code section of a distinct, second kind;

extracting a first set of language elements from the first script code section;

extracting a second set of language elements from the second script code section; and

comparing the first set of language elements and the second set of language elements with the definition module.

11. A computer program product, tangibly embodied in an information carrier, the computer program product comprising instructions operable to cause data processing equipment to:

receive a language-independent description of a computer program, the language-independent description comprising a definition module and an implementation module;

validate the language-independent description;

generate a language-dependent program from the language-independent description, the language-dependent program comprising an interface and a class; and

validate the language-dependent program.

12. The computer program product of claim 11, wherein the instructions to validate the language-independent description cause the data processing equipment to validate the syntax of the definition module and the implementation module.

13. The computer program product of claim 11, wherein the instructions to validate the language-dependent program cause the data processing equipment to compile the interface and the class.

14. The computer program product of claim 11 wherein the definition module and the implementation module are represented in a meta-language.

15. A computer program product, tangibly embodied in an information carrier, the computer program product comprising instructions operable to cause data processing equipment to:

receive a language-independent description of a computer program, the language-independent description comprising a definition module and an implementation module;

validate the language-independent description;

generate a language-dependent program from the language-independent description, the language-dependent program comprising a script code section; and

validate the language-dependent program.

16. The computer program product of claim 15, wherein the instructions to validate the language-dependent program cause the data processing equipment to:

extract language elements from the script code section; and

compare the extracted language elements with the definition module.

17. The computer program product of claim 16 wherein the instructions to extract the language elements cause the data processing equipment to generate a symbol table from the script code section.

18. The computer program product of claim 15, wherein the instructions to generate the language-dependent program cause the data processing equipment to:

generate language-dependent code comprising an interface and a class.

19. The computer program product of claim 15 wherein the instructions to validate the language-dependent program cause the data processing equipment to:

extract language elements from the script code section;

compare the extracted language elements with the definition module;

generate language-dependent code comprising an interface and a class; and

compile the interface and the class.

20. A computer program product, tangibly embodied in an information carrier, the computer program product comprising instructions operable to cause data processing equipment to:

receive a language-independent description of a computer program, the language-independent description comprising a definition module and an implementation module;

validate the language-independent description;

generate a first language-dependent program from the language-independent description, the first language-dependent program comprising a first script code section;

generate a second language-dependent program from the language-dependent description, the second language-dependent program comprising a second script code section of a distinct, second kind;

extract a first set of language elements from the first script code section;

extract a second set of language elements from the second script code section; and

compare the first set of language elements and the second set of language elements with the definition module.

21. An apparatus, comprising:

means for receiving a language-independent description of a computer program, the language-independent description comprising a definition module and an implementation module;

means for validating the language-independent description;

means for generating a language-dependent program from the language-independent description, the language-dependent program comprising an interface and a class; and

means for validating the language-dependent program.